

LAB GUIDE

Lab: Deploying a Docker based web application to Azure App Service

Learning Objectives

- How to build custom Docker images using Azure DevOps Hosted Linux agent
- How to push and store the Docker images in a private repository
- How to Deploy and run the images inside the Docker Containers

Pre-requisites

- Microsoft Azure Account: You'll need a valid and active Azure account for the Azure labs.
- You'll need an Azure DevOps account.

Length

40 minutes

Exercise 1: Create a new project in Azure DevOps

1. Log in [Azure DevOps](#)

Microsoft Azure

Contact Sales: 1-800-867-1389 Search My account Portal

Overview Solutions Products Documentation Pricing Training Marketplace Partners Support Blog More

Azure DevOps

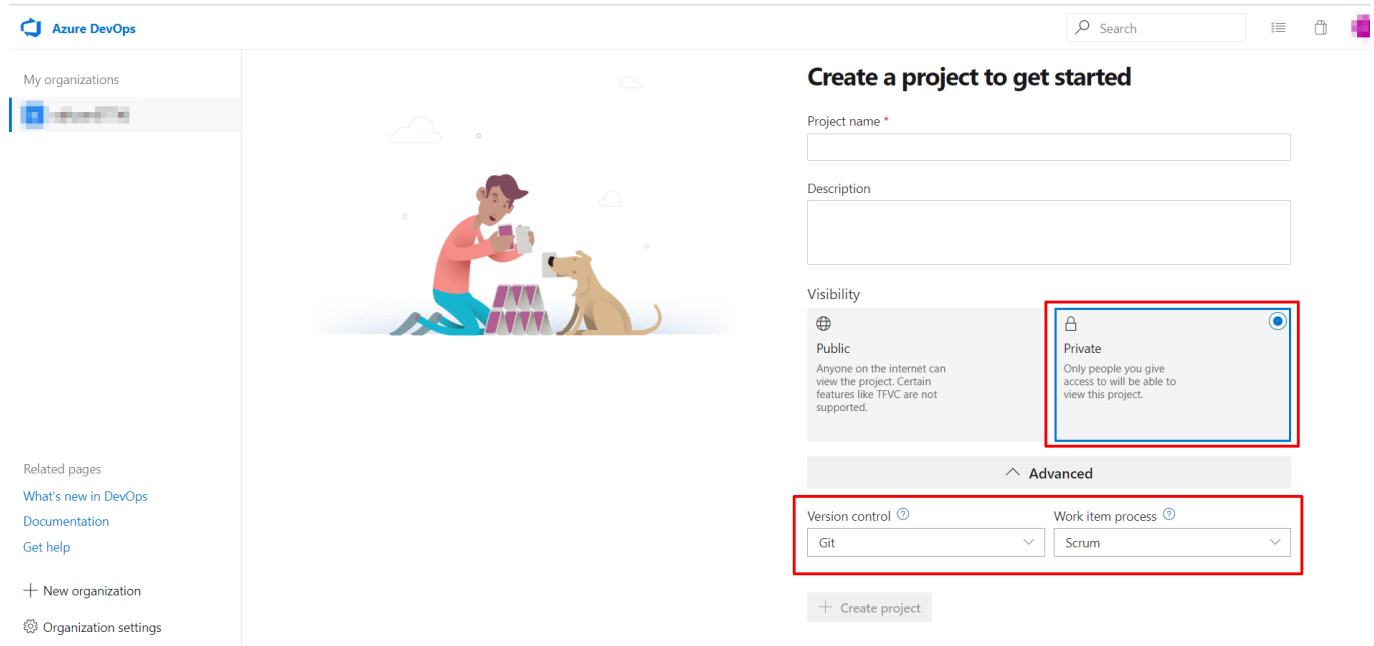
Plan smarter, collaborate better, and ship faster with a set of modern dev services.

[Start free](#)

Already have an account?
[Sign in to Azure DevOps >](#)

<https://go.microsoft.com/fwlink/?LinkId=2014881&WebUserId=082e7e96-2f96-436b-816e-1fe42eb9e90a&campaign=acom-azure-devops-se>

2. Set the name you want
3. In **visibility** select **Private**
4. Click on **Advanced**, in **Version control** select **GIT** and **Work item process** select **Scrum** then click on **+ Create project**



Azure DevOps

My organizations

Related pages

- What's new in DevOps
- Documentation
- Get help

+ New organization

Organization settings

Create a project to get started

Project name *

Description

Visibility

Public
Anyone on the internet can view the project. Certain features like TFVC are not supported.

Private
Only people you give access to will be able to view this project.

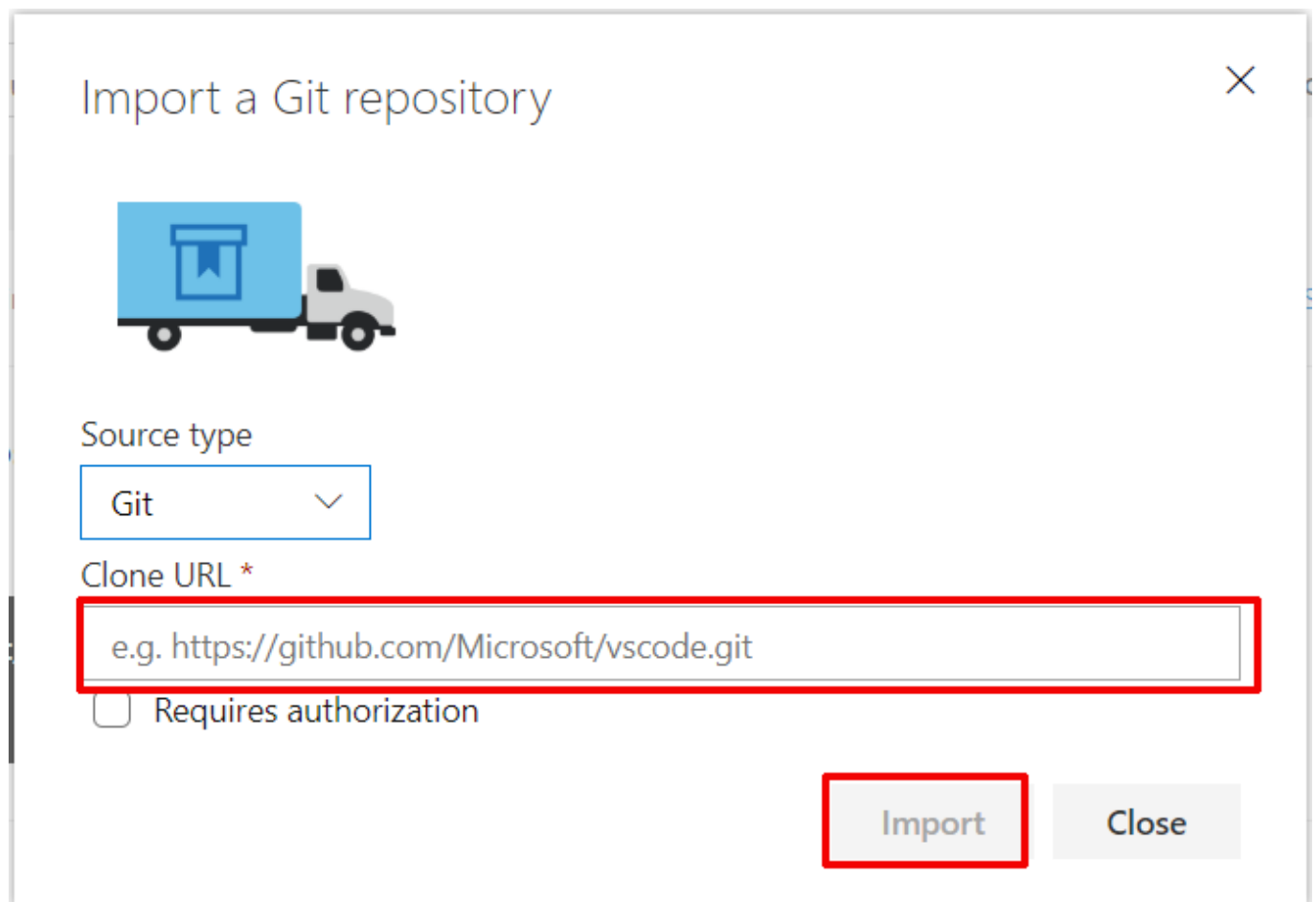
Advanced

Version control ?
Git


Work item process ?
Scrum

+ Create project

5. Once your project has been created click on **Repos** option
6. In Repos page you'll see many options to add some code, click on **Import** from **Or import a repository** option
7. in **Clone URL** option put this URL then click on import <https://github.com/MSTecs/Azure-DevDay-lab4-demoProject.git>



Import a Git repository



Source type

Git

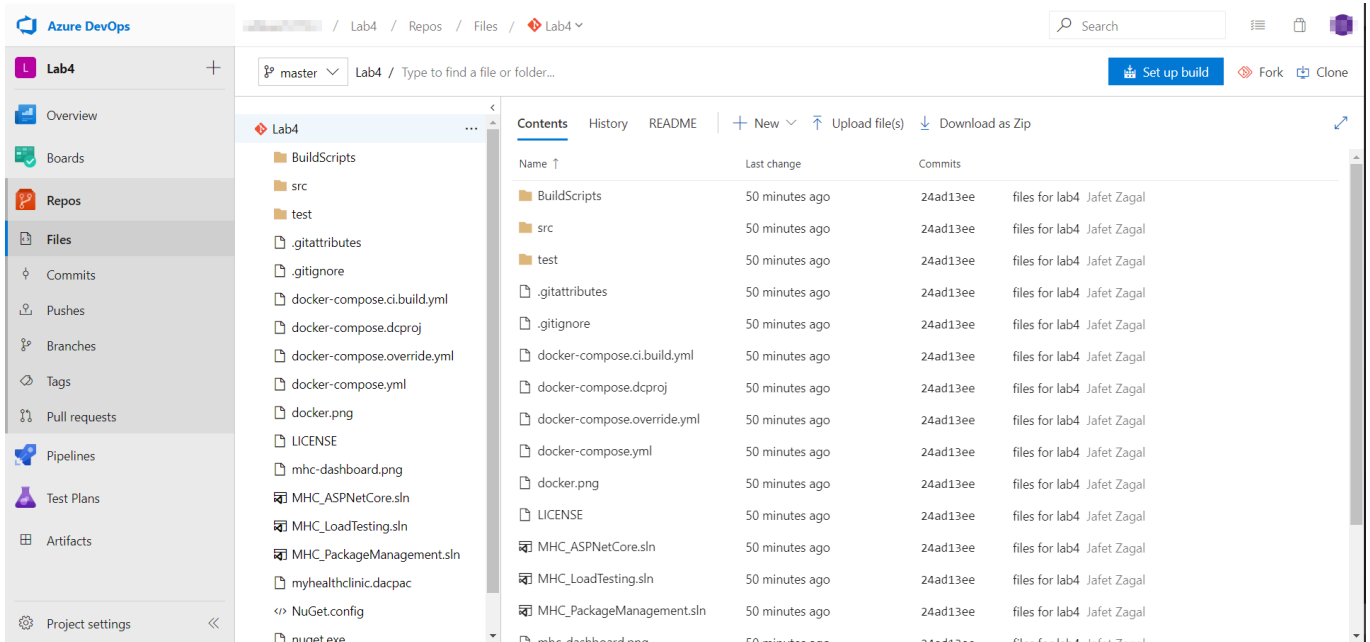
Clone URL *

e.g. <https://github.com/Microsoft/vscode.git>

☐ Requires authorization

Import Close

8. Now if you click again on files you will see the code in your page



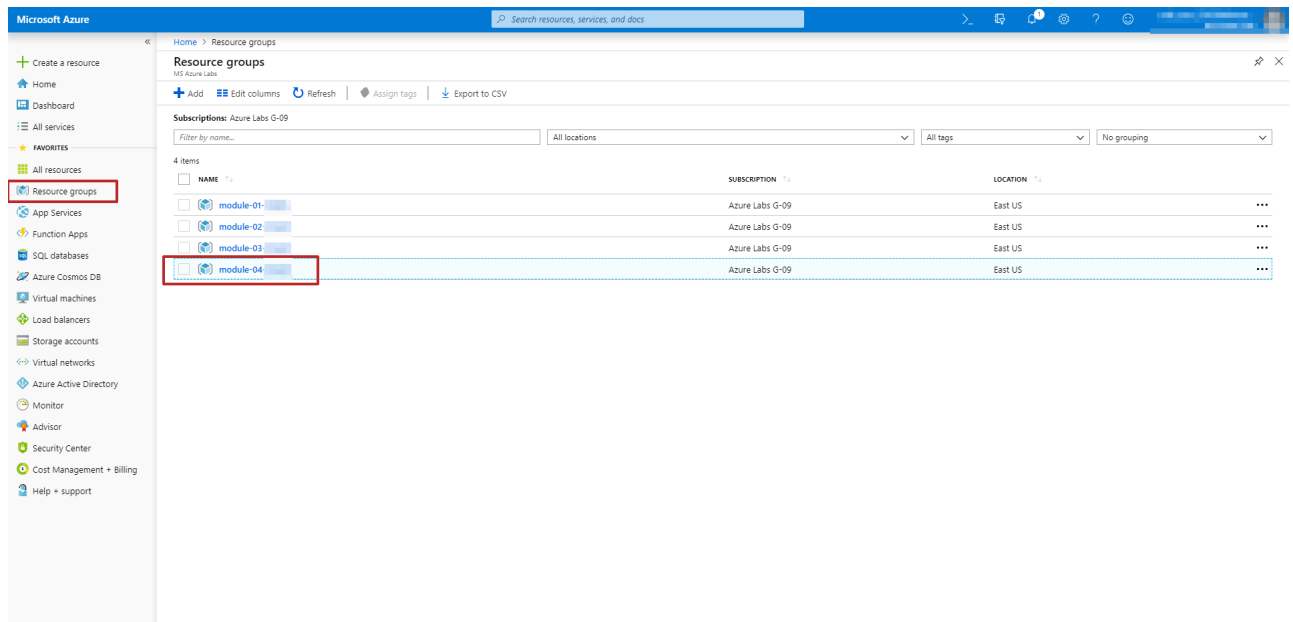
Exercise 2: Configure Continuous Integration

Task 1: Configure your basic Pipeline DevOps

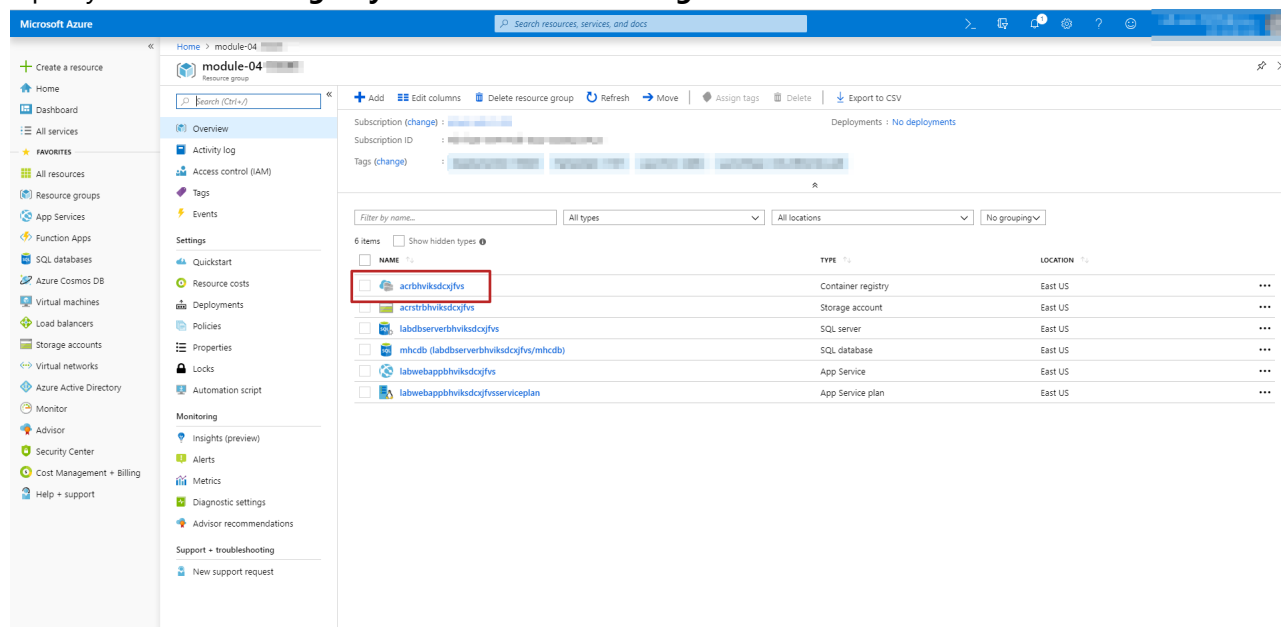
1. Log in portal Azure clicking here



2. Go to resources groups and select the resource named module-04-xxxxx Where xxxxx is the number obtained for the lab

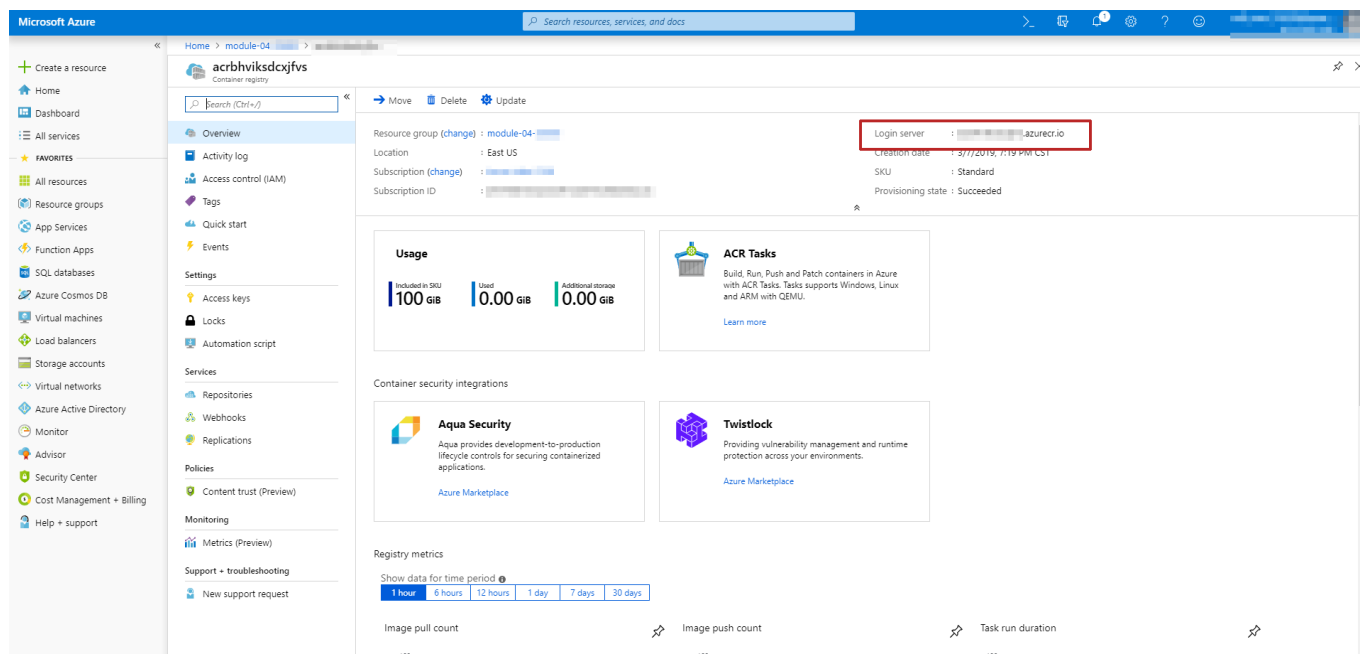


3. Open your **container registry** and take note of the **login server**



The screenshot shows the Microsoft Azure portal interface. The left sidebar contains navigation links for 'Create a resource', 'Home', 'Dashboard', 'All services', and 'FAVORITES'. The main area displays the 'module-04' resource group. The 'Overview' tab is active, showing a list of resources. The resource 'acrbitvhvskdcjvjs' is highlighted with a red box. The table below lists the resources in the group:

NAME	TYPE	LOCATION
acrbitvhvskdcjvjs	Container registry	East US
acrbitvhvskdcjvjs	Storage account	East US
labdbserverbitvhvskdcjvjs	SQL server	East US
mhcdb (labdbserverbitvhvskdcjvjs/mhcdb)	SQL database	East US
labwebappbitvhvskdcjvjs	App Service	East US
labwebappbitvhvskdcjvjs/serviceplan	App Service plan	East US



The screenshot shows the Microsoft Azure portal interface for the 'acrbitvhvskdcjvjs' container registry. The 'Overview' tab is active, displaying the registry's details and usage. The 'Login server' is highlighted with a red box. The 'Usage' section shows the registry's capacity and usage. The 'ACR Tasks' section provides information about building, running, pushing, and patching containers. The 'Container security integrations' section lists 'Aqua Security' and 'Twistlock'. The 'Registry metrics' section shows the image pull and push counts over time.

Resource group (change): module-04

Location: East US

Subscription (change): [Subscription ID]

Subscription ID: [Subscription ID]

Login server: [Login Server]

Creation date: 3/7/2019, 7:19 PM CST

SKU: Standard

Provisioning state: Succeeded

Usage

Used	Additional storage
100 GiB	0.00 GiB

ACR Tasks

Build, Run, Push and Patch containers in Azure with ACR Tasks. Tasks supports Windows, Linux and ARM with CERNU.

Learn more

Container security integrations

Aqua Security

Aqua provides development-to-production lifecycle controls for securing containerized applications.

Azure Marketplace

Twistlock

Providing vulnerability management and runtime protection across your environments.

Azure Marketplace

Registry metrics

Show data for time period

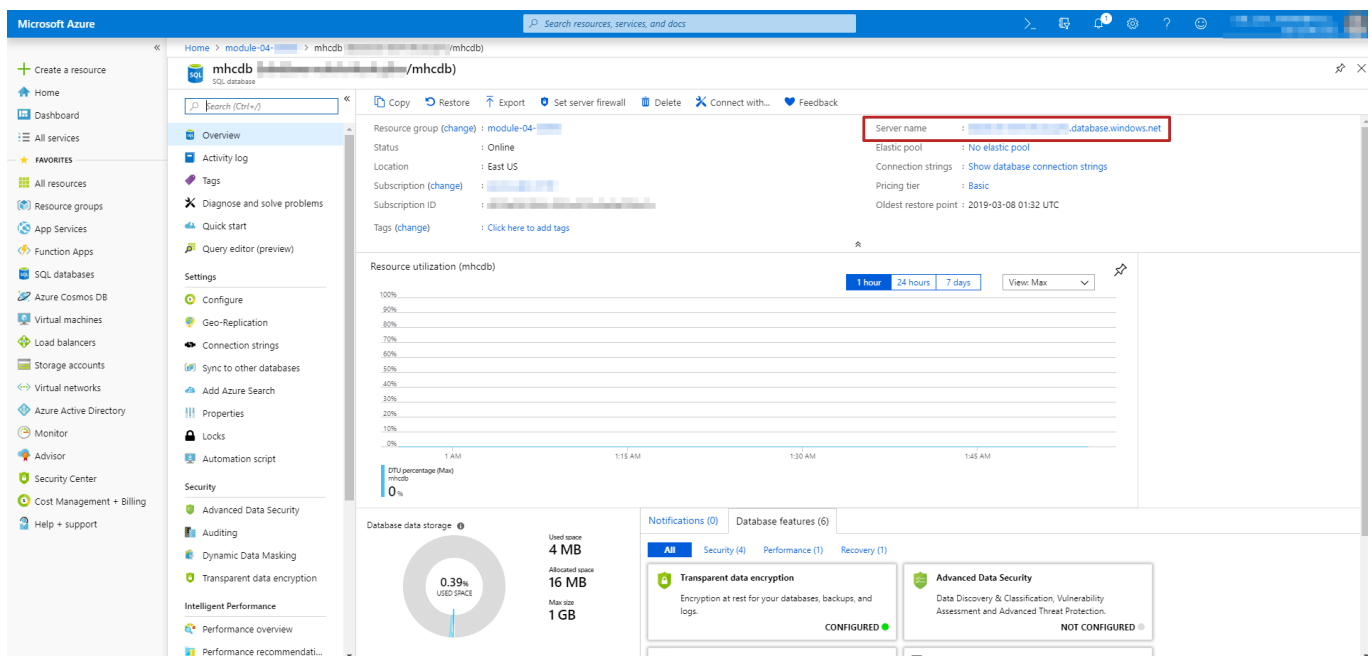
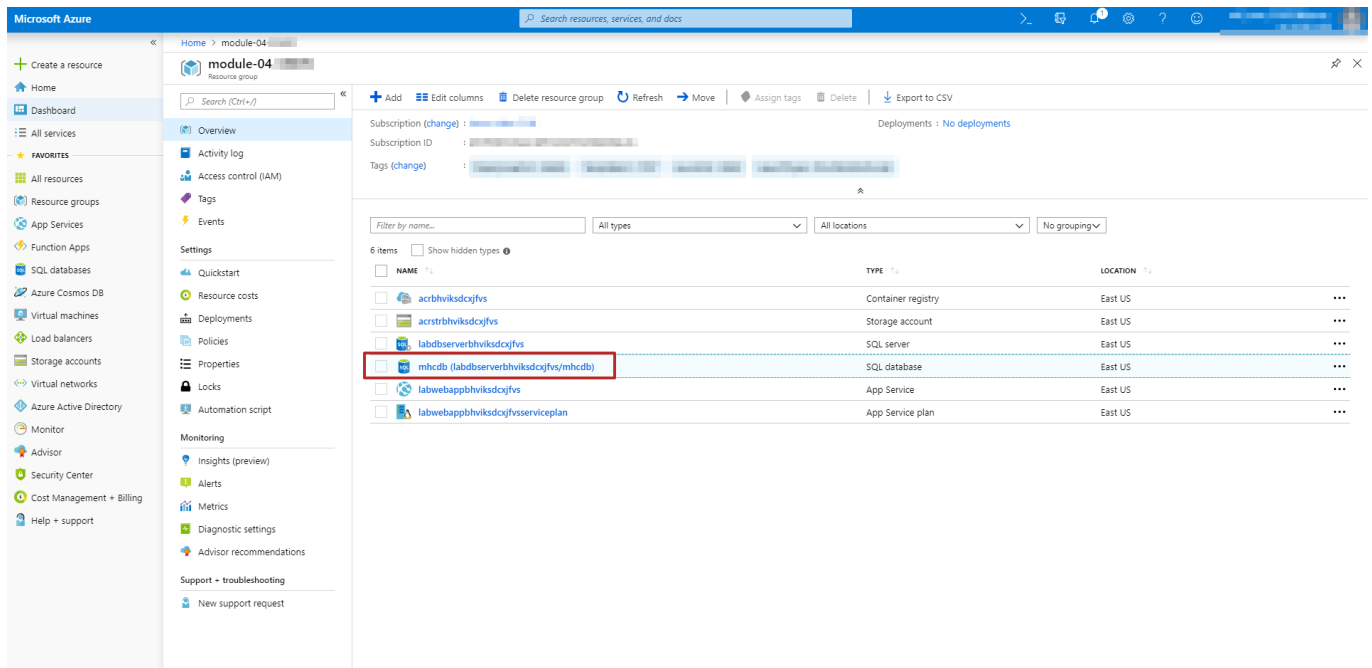
1 hour 6 hours 12 hours 1 day 7 days 30 days

Image pull count

Image push count

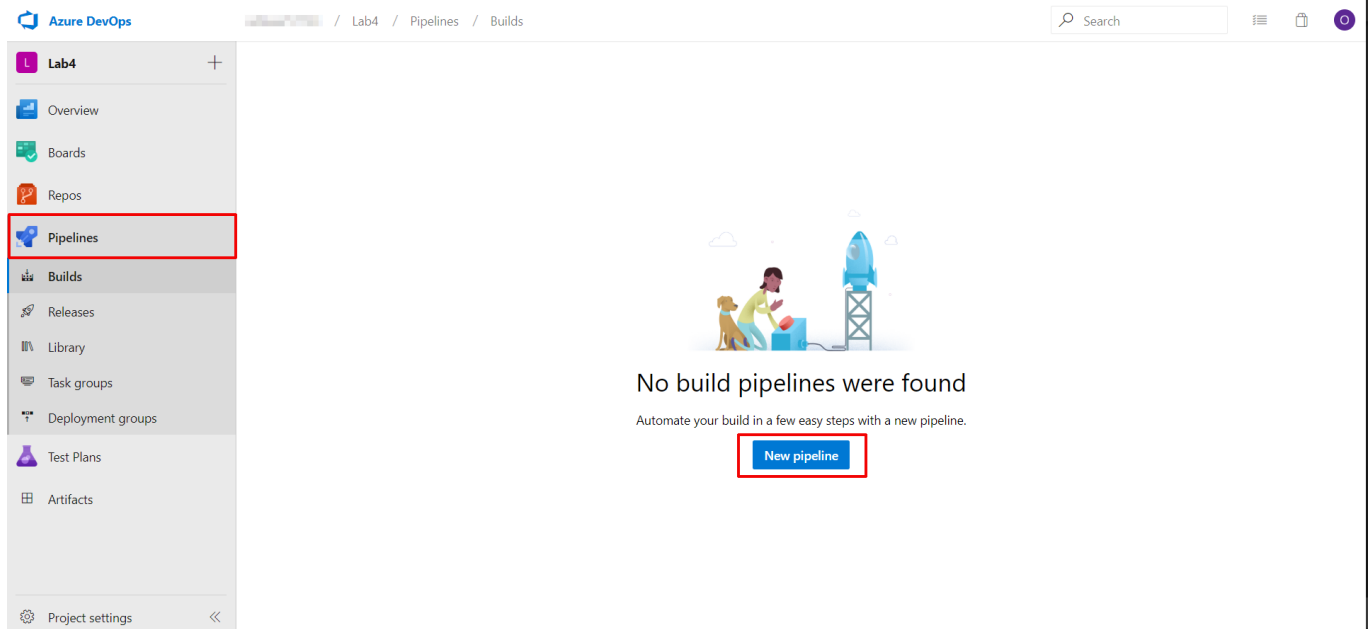
Task run duration

4. Go back to your resources open your **SQL database** and take note of the **Server name**



5. Return to [Azure DevOps](#)

6. Navigate to the **Builds** option under the **Pipelines** tab and select new pipeline



7. Select **Use the classic editor** option

Connect Select Configure Create pipeline

New pipeline

Where is your code?



Azure Repos Git YAML

Free private Git repositories, pull requests, and code search



Bitbucket Cloud

Hosted by Atlassian



GitHub YAML

Home to the world's largest community of developers



GitHub Enterprise Server YAML

The self-hosted version of GitHub Enterprise



Other Git

Any Internet-facing Git repository




Subversion

Centralized version control by Apache

[Use the classic editor](#) to create a pipeline without YAML.


8. Select **Azure Repos Git** and select your project and click on Continue





Select your repository


Tell us where your sources are.
You can customize how to get these sources from the repository later.


Select a source


 Azure Repos Git

 GitHub

 GitHub Enterprise Server

 Subversion

 Bitbucket Cloud

 External Git

Team project

Lab4

Repository


Lab4

Default branch for manual and scheduled builds

master

Continue


9. Select the **Empty job** option



Choose a template


Choose a template that builds your kind of app.
Don't worry if it's not an exact match;
you can add and customize the tasks later.

Select a template


Or start with an  **Empty job**


Search


Configuration as code


 **YAML**
Looking for a better experience to configure your pipelines using YAML files? Try the new YAML pipeline creation experience. [Learn more](#)


Featured


 **.NET Desktop**
Build and test a .NET or Windows classic desktop solution.

 **Android**
Build, test, sign, and align an Android APK.

 **ASP.NET**
Build and test an ASP.NET web application.

 **Azure Web App for ASP.NET**
Build, package, test, and deploy an ASP.NET Azure Web App.

 **Docker container**
Build a Docker image and push it to a container registry.

 **Maven**
Build and test a Java project with Apache Maven.

10. Click on the **Agent job 1** and change the **display name** to Docker

11. On **Execution Plan** Section set 1 on **Job cancel timeout** input

Lab4-CI

Tasks Variables Triggers Options Retention History Save & queue Discard Summary Queue

Pipeline Build pipeline

Get sources Lab4 master

Agent job 1 Run on agent

Display name * Agent job 1

Agent selection ^

Agent pool | Manage <inherit from pipeline>

Demands

Name	Condition	Value
------	-----------	-------

+ Add

Execution plan ^

Parallelism

☒ None ☐ Multi-configuration ☐ Multi-agent

Timeout * 0

Job cancel timeout * 0

Your configuration should look like this

Lab4-CI

Tasks Variables Triggers Options Retention History Save & queue Discard Summary Queue

Pipeline Build pipeline

Get sources Lab4 master

Docker Run on agent

Agent job

Display name * Docker

Agent selection ^

Agent pool | Manage <inherit from pipeline>

Demands

Name	Condition	Value
------	-----------	-------

+ Add

Execution plan ^

Parallelism

☒ None ☐ Multi-configuration ☐ Multi-agent

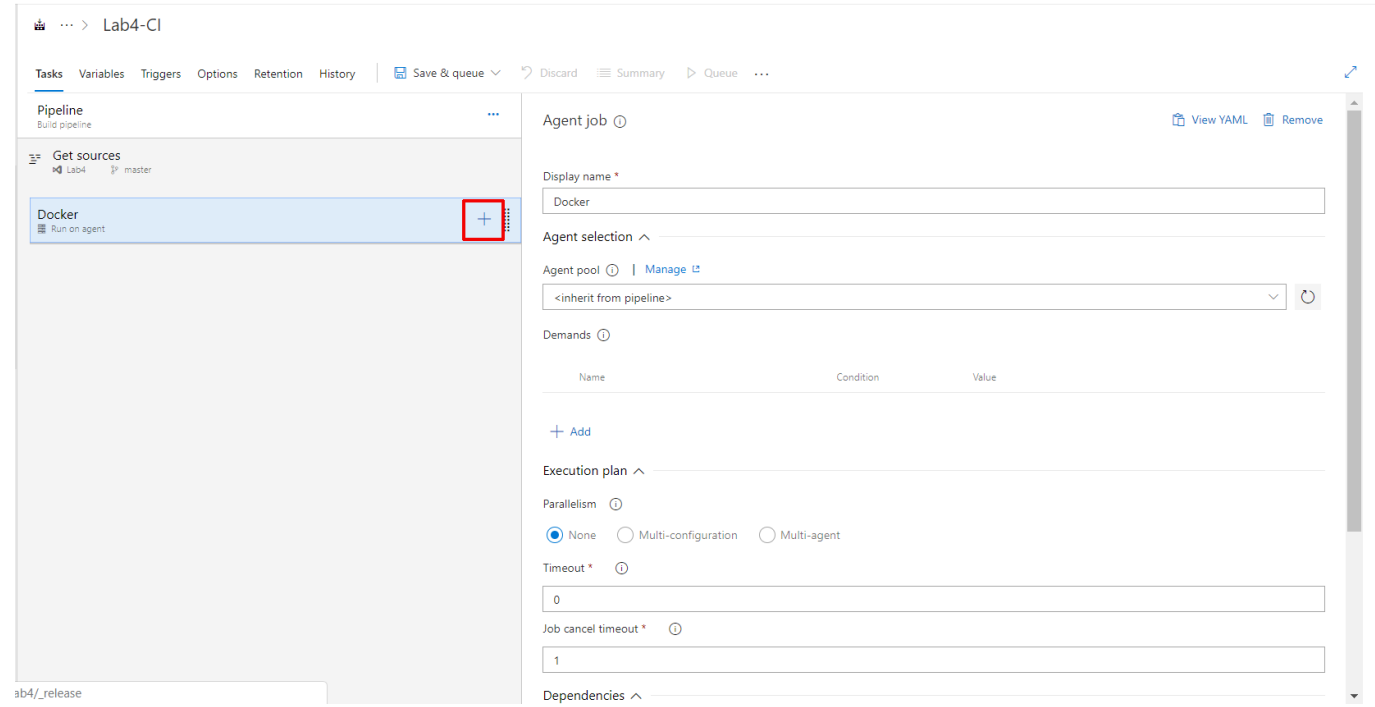
Timeout * 0

Job cancel timeout * 1

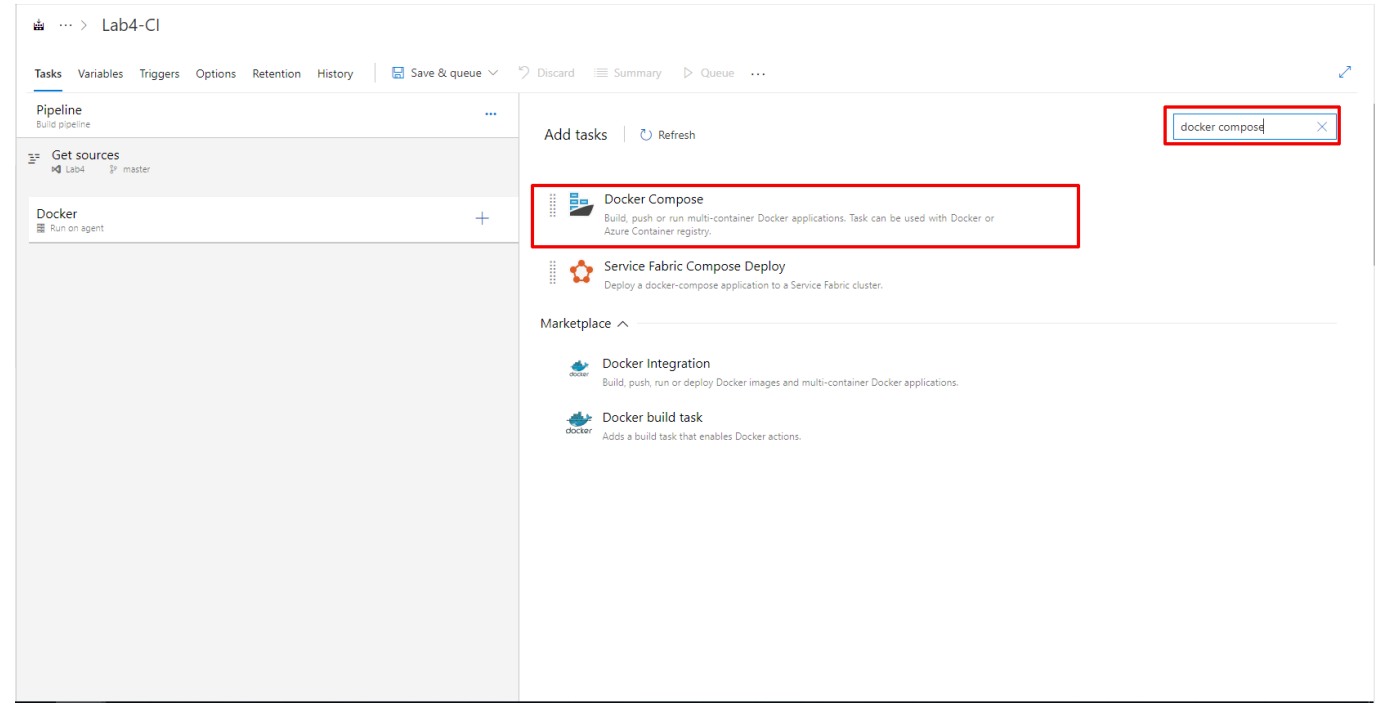
Dependencies ^

Task 2: create your Run Services

1. Click on plus button + on your Docker agent



2. Search for **docker compose** and select the first option



3. Set **Run services** on **Display name** input and select your azure subscription

Lab4-CI

Tasks Variables Triggers Options Retention History Save & queue Discard Summary Queue

Pipeline Build pipeline

Get sources Lab4 master

Docker Run on agent

Run services Some settings need attention

Docker Compose

Task version 0.*

Display name * Run services

Container Registry Type * Azure Container Registry

Azure subscription | Manage

Click Authorize to configure an Azure service connection

Azure Container Registry

Docker Compose File * **/docker-compose.yml

Additional Docker Compose Files

4. Click **Advanced options** as the image shown

Lab4-CI

Tasks Variables Triggers Options Retention History Save & queue Discard Summary Queue

Pipeline Build pipeline

Get sources Lab4 master

Docker Run on agent

Run services Some settings need attention

Docker Compose

Task version 0.*

Display name * Run services

Container Registry Type * Azure Container Registry

Azure subscription | Manage

Click Authorize to configure an Azure service connection

Azure Container Registry

Docker Compose File * **/docker-compose.yml

Additional Docker Compose Files

Advanced options

5. Click on **use the full version of the service connection dialog** option

×

Add an Azure Resource Manager service connection

☒ Service Principal Authentication ☐ Managed Identity Authentication

Connection name

Scope level

Subscription

▼

Subscription

Resource Group

▼

Subscriptions listed are from Azure Cloud

A new Azure service principal will be created and assigned with "Contributor" role, having access to all resources within the subscription. Optionally, you can select the Resource Group to which you want to limit access.

If your subscription is not listed above, or your organization is not backed by Azure Active Directory, or to specify an existing service principal, [use the full version of the service connection dialog](#).

☒ Allow all pipelines to use this connection.

OK

Close

6. Set `serviceConnection` on **Connection name** input

7. Put your **Service Principal Details** given at the beginning of the lab and click in **Ok**

- **Application/Client Id**
- **Application Secret Key**

×

Add an Azure Resource Manager service connection

☒ Service Principal Authentication ☐ Managed Identity Authentication

Connection name

ServiceConnection

Environment

AzureCloud

Scope level

Subscription

Subscription ID

Subscription name

Service principal client ID

☒ Service principal key ☐ Certificate

Service principal key

.....

Tenant ID

Connection: Not verified

Verify connection

For help on creating an Azure service principal, see [service connections](#).

OK

Close

8. Select your container registry

The screenshot displays the Azure DevOps web interface for configuring a pipeline task. On the left, the 'Pipeline' sidebar shows the 'Run services' task selected under the 'Docker' category. The main panel is titled 'Docker Compose' and contains the following configuration fields:

- Task version:** 0.*
- Display name:** Run services
- Container Registry Type:** Azure Container Registry
- Azure subscription:** ServiceConnection (highlighted with a red box)
- Azure Container Registry:** [Redacted] (highlighted with a red box)
- Docker Compose File:** **/docker-compose.yml
- Additional Docker Compose Files:** (Empty text area)
- Environment Variables:** (Empty text area)

Buttons for 'Link settings', 'View YAML', and 'Remove' are located in the top right corner of the task configuration panel.

9. Click on ellipsis button (...) and search the file **docker-compose.ci.build.yml**

The screenshot shows the Azure DevOps interface for a pipeline named 'Lab4-CI'. The left sidebar shows the pipeline structure with 'Get sources' and 'Run services' tasks. The 'Run services' task is selected, and its configuration is shown on the right. The configuration includes fields for 'Task version' (0.*), 'Display name' (Run services), 'Container Registry Type' (Azure Container Registry), 'Azure subscription' (ServiceConnection), 'Azure Container Registry' (a masked value), 'Docker Compose File' (**/docker-compose.yml), and 'Additional Docker Compose Files'. A red box highlights the ellipsis button next to the 'Docker Compose File' field. Below the configuration fields is an 'Environment Variables' section.

Select path

- Lab4
 - BuildScripts
 - src
 - test
 - .gitattributes
 - .gitignore
 - docker-compose.ci.build.yml**
 - docker-compose.dcpj
 - docker-compose.override.yml
 - docker-compose.yml
 - docker.png

OK

Cancel

- Go down in your **Run services** configuration and select **Run service images** on **Action** option
- Uncheck the **Run in Background** option

12. In **Output Variables** set as **Task1** in **Reference name** input

The screenshot shows the Jenkins configuration page for a task named 'Run service images'. The left sidebar shows the pipeline structure with 'Run services' selected. The main configuration area has the following settings:

- Project Name:** \$(Build.Repository.Name)
- Qualify Image Names:** ☒
- Action:** Run service images
- Build Images:** ☒
- Run in Background:** ☐
- Abort on Container Exit:** ☒
- Advanced Options:** (collapsed)
- Control Options:** (collapsed)
- Output Variables:** (expanded)
 - Reference name:** Task1
 - Variables list:** There are no output variables associated with this task [more information](#)

Task 3: create your Build services

1. Click on plus button (+) and search again for Docker compose

The top screenshot shows the 'Run services' task configuration in the Azure DevOps pipeline editor. The task is named 'Run services' and uses the 'Docker Compose' action. The 'Project Name' is set to '\$(Build.Repository.Name)'. The 'Action' is 'Run service images'. The 'Build Images' checkbox is checked. The 'Advanced Options' section is expanded, showing 'Control Options' and 'Output Variables'. The 'Reference name' is set to 'Task1'.

The bottom screenshot shows the 'Add tasks' section of the pipeline editor. A search bar at the top right contains the text 'docker compose'. The 'Docker Compose' task is highlighted in a red box. The task description is: 'Build, push or run multi-container Docker applications. Task can be used with Docker or Azure Container registry.' Below this, the 'Service Fabric Compose Deploy' task is listed, followed by the 'Marketplace' section which includes 'Docker Integration' and 'Docker build task'.

2. Set **Build services** on **Display name** input
3. Select your **ServiceConnection** *previously created* on **Azure subscription**
4. Select your **Azure Container registry**
5. Search the **docker-compose.yml** file clicking on the Ellipsis button (...), selecting the file and clicking **Ok**
6. Put this line **DOCKER_BUILD_SOURCE=** on your **Environment Variables**

Your config should look like this

Lab4-CI

Tasks Variables Triggers Options Retention History Save & queue Discard Summary Queue

Pipeline Build pipeline

Get sources Lab4 master

Docker Run on agent

Run services Docker Compose

Build services Docker Compose Some settings need attention

Display name * Build services

Container Registry Type * Azure Container Registry

Azure subscription | Manage ServiceConnection

Azure Container Registry

Docker Compose File * docker-compose.yml

Additional Docker Compose Files

Environment Variables DOCKER_BUILD_SOURCE=

Project Name

7. Go down and select **Build service images** on **Action** dropdown

8. Set as **Task2** on **Reference name** on **Output Variables** section

Lab4-CI

Tasks Variables Triggers Options Retention History Save & queue Discard Summary Queue

Pipeline Build pipeline

Get sources Lab4 master

Docker Run on agent

Run services Docker Compose

Build services Docker Compose

\$(Build.Repository.Name)

☒ Qualify Image Names

Action * Build service images

Additional Image Tags

☐ Include Source Tags

☐ Include Latest Tag

Advanced Options

Control Options

Output Variables

Reference name Task2

Variables list

There are no output variables associated with this task [more information](#)

Task 4: Create your Push services

1. Click on plus button (+) and search again for Docker compose

The first screenshot shows the 'Build services' task configuration in the 'Lab4-CI' pipeline. The task is selected, and the 'Build service images' action is chosen. The 'Reference name' is set to 'Task2'. A red box highlights the '+' icon next to the 'Build services' task in the left sidebar.

The second screenshot shows the 'Add tasks' dialog with 'docker compose' entered in the search bar. The 'Docker Compose' task is highlighted in the results list. A red box highlights the search bar and the 'Docker Compose' task entry.

2. Set **Push services** on **Display name** input
3. Select your **ServiceConnection** *previously created* on **Azure subscription**
4. Select your **Azure Container registry**
5. Search the **docker-compose.yml** file clicking on the Ellipsis button (...), selecting the file and clicking **Ok**
6. Put this line **DOCKER_BUILD_SOURCE=** on your **Environment Variables**

Your config should look like this

Lab4-CI

Tasks Variables Triggers Options Retention History Save & queue Discard Summary Queue

Pipeline Build pipeline

Get sources Lab4 master

Docker Run on agent

Run services Docker Compose

Build services Docker Compose

Push services Docker Compose Some settings need attention

Display name * Push services

Container Registry Type * Azure Container Registry

Azure subscription | Manage ServiceConnection

Azure Container Registry (selected registry)

Docker Compose File * docker-compose.yml

Additional Docker Compose Files

Environment Variables DOCKER_BUILD_SOURCE=

Project Name

7. Go down and select **Push service images** on **Action** dropdown

8. Set as **Task3** on **Reference name** on **Output Variables** section

Lab4-CI

Tasks Variables Triggers Options Retention History Save & queue Discard Summary Queue

Pipeline Build pipeline

Get sources Lab4 master

Docker Run on agent

Run services Docker Compose

Build services Docker Compose

Push services Docker Compose

\$(Build.Repository.Name)

☒ Qualify Image Names

Action * Push service images

Additional Image Tags

☐ Include Source Tags

☐ Include Latest Tag

Advanced Options

Control Options

Output Variables

Reference name Task3

Variables list

There are no output variables associated with this task more information

Task 5: create your Publish Artifact

1. Click on plus button (+) and search for Publish build artifacts

Lab4-CI

Tasks Variables Triggers Options Retention History | Save & queue Discard Summary Queue ...

Pipeline Build pipeline

Get sources Lab4 master

Docker Run on agent

Run services Docker Compose

Build services Docker Compose

Push services Docker Compose

\$(Build.Repository.Name)

☒ Qualify Image Names

Action * Push service images

Additional Image Tags

☐ Include Source Tags

☐ Include Latest Tag

Advanced Options

Control Options

Output Variables

Reference name Task3

Variables list

There are no output variables associated with this task [more information](#)

Lab4-CI

Tasks Variables Triggers Options Retention History | Save & queue Discard Summary Queue ...

Pipeline Build pipeline

Get sources Lab4 master

Docker Run on agent

Run services Docker Compose

Build services Docker Compose

Push services Docker Compose

Add tasks Refresh

publish build artifacts

Publish Build Artifacts

Publish build artifacts to Azure Pipelines/TFS or a file share

Deprecated tasks

2. Set **Publish Artifact** on **Display name** input
3. Set **myhealthclinic.dacpac** on **Path to publish** input
4. Set **dacpac** on **Artifact name** input
5. Set **PublishBuildArtifacts2** as your **Reference name** in the **Output variables**

Lab4-CI

Tasks Variables Triggers Options Retention History | Save & queue Discard Summary Queue ...

Pipeline Build pipeline

Get sources Lab4 master

Docker Run on agent

Run services Docker Compose

Build services Docker Compose

Push services Docker Compose

Publish Artifact Publish Build Artifacts

Display name * Publish Artifact

Path to publish * myhealthclinic.dacpac

Artifact name * dacpac

Artifact publish location * Azure Pipelines/TFS

Control Options

Output Variables

Reference name PublishBuildArtifacts2

Variables list

There are no output variables associated with this task [more information](#)

Task 6: Set your variables

1. Click on **Variables** tab
2. Create this variables clicking on **+ Add** button

Name: **BuildConfiguration** Value: **Release** check the *settable at queue time* Name: **BuildPlatform** Value: **Any CPU**

Lab4-CI

Tasks Variables Triggers Options Retention History | Save & queue Discard Summary Queue ...

Pipeline variables

Variable groups

Predefined variables

Name ↑	Value	Settable at queue time
BuildConfiguration	Release	<input checked="" type="checkbox"/>
BuildPlatform	Any CPU	<input checked="" type="checkbox"/>
system.collectionId	6548c617-84e7-42eb-9a62-270617775037	<input checked="" type="checkbox"/>
system.debug	false	<input checked="" type="checkbox"/>
system.definitionId	< No pipeline id yet >	
system.teamProject	Lab4	

+ Add

Task 7: config your triggers

1. Click on **Triggers** tab
2. Check the **Enable continuous integration** option

The screenshot shows the 'Lab4-CI' configuration page in Azure DevOps, specifically the 'Triggers' tab. The 'Continuous integration' section is active, showing 'Lab4' as 'Enabled'. Below this, there are sections for 'Scheduled' (no builds scheduled) and 'Build completion' (build when another build completes). On the right side, the 'Lab4' configuration is shown with 'Enable continuous integration' checked and highlighted with a red box. There is also an unchecked option for 'Batch changes while a build is in progress'. Below that, 'Branch filters' are configured with 'Type' set to 'Include' and 'Branch specification' set to '*/master'. 'Path filters' are also present with an 'Add' button.

Task 8: config your options

1. Click on **Options** tab
2. Put this line `$(date:yyyyMMdd)$(rev:.r)` on **Build number format**
3. Set as `0` on **Build job timeout in minutes**

The screenshot shows the 'Lab4-CI' configuration page in Azure DevOps, specifically the 'Options' tab. The 'Build properties' section is visible, with a 'Description' field and a 'Build number format' field containing the text `$(date:yyyyMMdd)$(rev:.r)`, which is highlighted with a red box. Below this, there are radio buttons for 'The new build request is processing', with 'Enabled' selected. There are also toggle switches for 'Automatically link new work in this build' and 'Create work item on failure', both of which are currently disabled. On the right side, the 'Build job' configuration is shown, with 'Build job authorization scope' set to 'Project collection', 'Build job timeout in minutes' set to `0` (highlighted with a red box), and 'Build job cancel timeout in minutes' set to `5`. The 'Demands' section is also visible, with a table for 'Name' and 'Condition'.

Task 9: config your Retention

1. Click on **Retention** tab
2. Click **Keep for 10 days, 1 good build**

3. Click on Delete

The screenshot shows the Jenkins Lab4-CI configuration page with the 'Retention' tab active. The 'Policies' section on the left lists two policies: 'Keep for 10 days, 1 good build' (highlighted with a red box) and 'Keep for 30 days, 10 good builds'. The 'Settings' section on the right shows branch filters and cleanup options. The 'Delete' button in the top right corner of the settings panel is also highlighted with a red box.

Your Retention should look like this

The screenshot shows the Jenkins Lab4-CI configuration page with the 'Retention' tab active. The 'Policies' section on the left lists two policies: 'Keep for 10 days, 1 good build' and 'Keep for 30 days, 10 good builds' (highlighted with a blue box). The 'Settings' section on the right shows branch filters and cleanup options. The 'Days to keep' is set to 30 and 'Minimum to keep' is set to 10.

Task 10: Set your host

1. Click on **Tasks** tab
2. Click on Pipelines
3. Select **Hosted Ubuntu 1604** option from **Agent Pool** dropdown

The screenshot shows the Azure DevOps 'Lab4-CI' pipeline configuration page. The 'Tasks' tab is active and highlighted with a red box. Below it, the 'Pipeline' section is highlighted with a red box. The 'Name' field is set to 'Lab4-CI'. The 'Agent pool' dropdown is set to 'Hosted Ubuntu 1604' and is also highlighted with a red box. The left sidebar shows various task categories like 'Get sources', 'Docker', 'Run services', 'Build services', 'Push services', and 'Publish Artifact'.

4. Click on **Save & queue** dropdown and select the **save** option
5. Put any comment you want

Exercise 3: configure your Continuous Delivery

Task 1: create the pipeline

1. Navigate to the **Releases** section under the **Pipelines** tab and select **New pipeline** and select **Empty job**

The image shows two screenshots of the Azure DevOps web interface. The top screenshot shows the 'Releases' section in the left-hand navigation pane, which is highlighted with a red box. The main area displays a message: 'No release pipelines found' with a subtext 'Automate your release process in a few easy steps with a new pipeline' and a 'New pipeline' button, also highlighted with a red box. The bottom screenshot shows the 'New release pipeline' wizard. The 'Select a template' step is highlighted with a red box, showing a search bar and a list of featured templates. The templates include: 'Azure App Service deployment', 'Deploy a Java app to Azure App Service', 'Deploy a Node.js app to Azure App Service', 'Deploy a PHP app to Azure App Service and Azure Database for MySQL', 'Deploy a Python app to Azure App Service and Azure database for MySQL', 'Deploy to a Kubernetes cluster', and 'IIS website and SQL database deployment'. The 'Stage 1' section shows a 'Select a template' button.

2. Set **Dev** on **Stage name** option

All pipelines > New release pipeline

Pipeline Tasks Variables Retention Options History

Artifacts | + Add

Stages | + Add

Dev
1 job, 0 task

Stage

Dev

Properties

Name and owners of the stage

Stage name
Dev

Stage owner

3. Click on **Add an artifact** and set the inputs as below

All pipelines > New release pipeline

Pipeline Tasks Variables Retention Options History

Artifacts | + Add

Stages | + Add

Dev
2 jobs, 2 tasks

Add an artifact

Source type

Build Azure Repos ... GitHub TFVC

3 more artifact types

Project *
Lab4

Source (build pipeline) *
Lab4-CI

Default version *
Latest

Source alias *
Lab4-CI

The artifacts published by each version will be available for deployment in release pipelines. The latest successful build of **Lab4-CI** published the following artifacts: **dacpac**.

Add

4. Set Continuous deployment trigger clicking on the ray icon and switching to enable the Continuous deployment trigger

The screenshot shows the 'New release pipeline' configuration in Azure DevOps. In the 'Artifacts' section, the 'Lab4-CI' artifact is highlighted with a red box. In the 'Stages' section, the 'Dev' stage is shown with '2 jobs, 2 tasks'. On the right, the 'Continuous deployment trigger' is set to 'Enabled' (highlighted with a red box), and the 'Pull request trigger' is set to 'Disabled'.

Task 2: Config your DB deployment

1. Click on **Tasks** tab
2. Click on **Agent job**
3. set **DB Deployment** on **Display name** input
4. Create this variable clicking on **+ Add** button **Name:** `sqlpackage` **Condition:** `exists`
5. Click on plus button (+) and search for `azure SQL database deployment`
6. Select the first option then add

The screenshot shows the 'Add tasks' interface in Azure DevOps. The search bar contains the text 'azure SQL databas...' (highlighted with a red box). The first result, 'Azure SQL Database Deployment', is highlighted with a red box, and the 'Add' button next to it is also highlighted with a red box.

7. Select your **Azure SQL DacpacTask**
8. Put `Execute Azure SQL: DacpacTask` as the **Display name**
9. Select your **ServiceConnection** on **Azure Subscription**
10. Set `$(SQLserver)` on **Azure SQL Server** input
11. Set `$(DatabaseName)` on **Database** input
12. Set `$(SQLadmin)` (with a blank space at the beginning) on **Login** input
13. Set `$(Password)` on **Password** input

All pipelines > New release pipeline

Pipeline Tasks Variables Retention Options History

Dev
Deployment process

DB deployment
Run on agent

Execute Azure SQL : DacpacTask
Some settings need attention

ServiceConnection

SQL Database ^

Authentication Type * ①
SQL Server Authentication

Azure SQL Server * ①
\$(SQLServer)

Database * ①
\$(DatabaseName)

Login * ①
\$(SQLadmin)

Password * ①
\$(Password)

Deployment Package ^

Deploy type *
SQL DACPAC File

Action * ①
Publish

- Go down and set `$(System.DefaultWorkingDirectory)/**/*.dacpac` as your **DACPAC File** on **Deployment Package** section
- Set `SqlAzureDacpacDeployment1` as your **Reference name** on **Output Variables** section

All pipelines > New release pipeline

Pipeline Tasks Variables Retention Options History

Dev
Deployment process

DB deployment
Run on agent

Execute Azure SQL : DacpacTask
Azure SQL Database Deployment

Action * ①
Publish

DACPAC File * ①
\$(System.DefaultWorkingDirectory)/**/*.dacpac

Publish Profile ①

Additional SqlPackage.exe Arguments ①

Firewall ^

Control Options ^

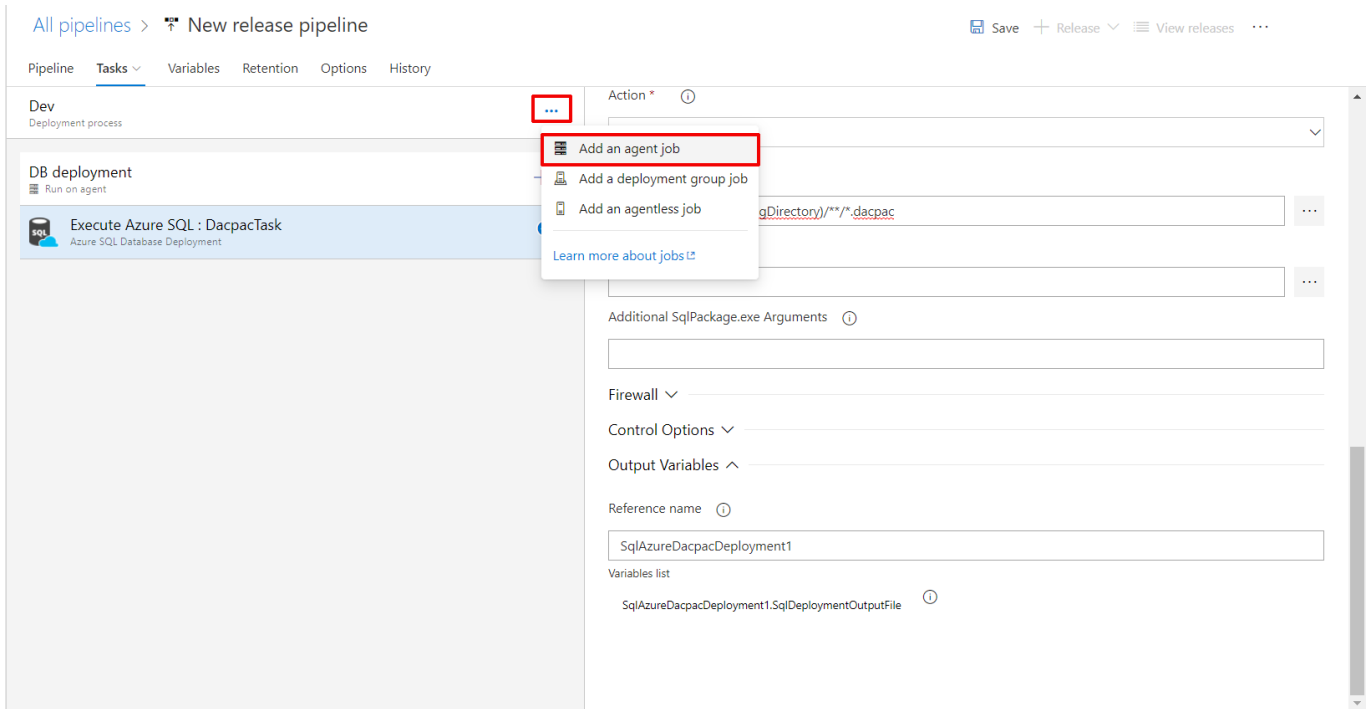
Output Variables ^

Reference name ①
SqlAzureDacpacDeployment1

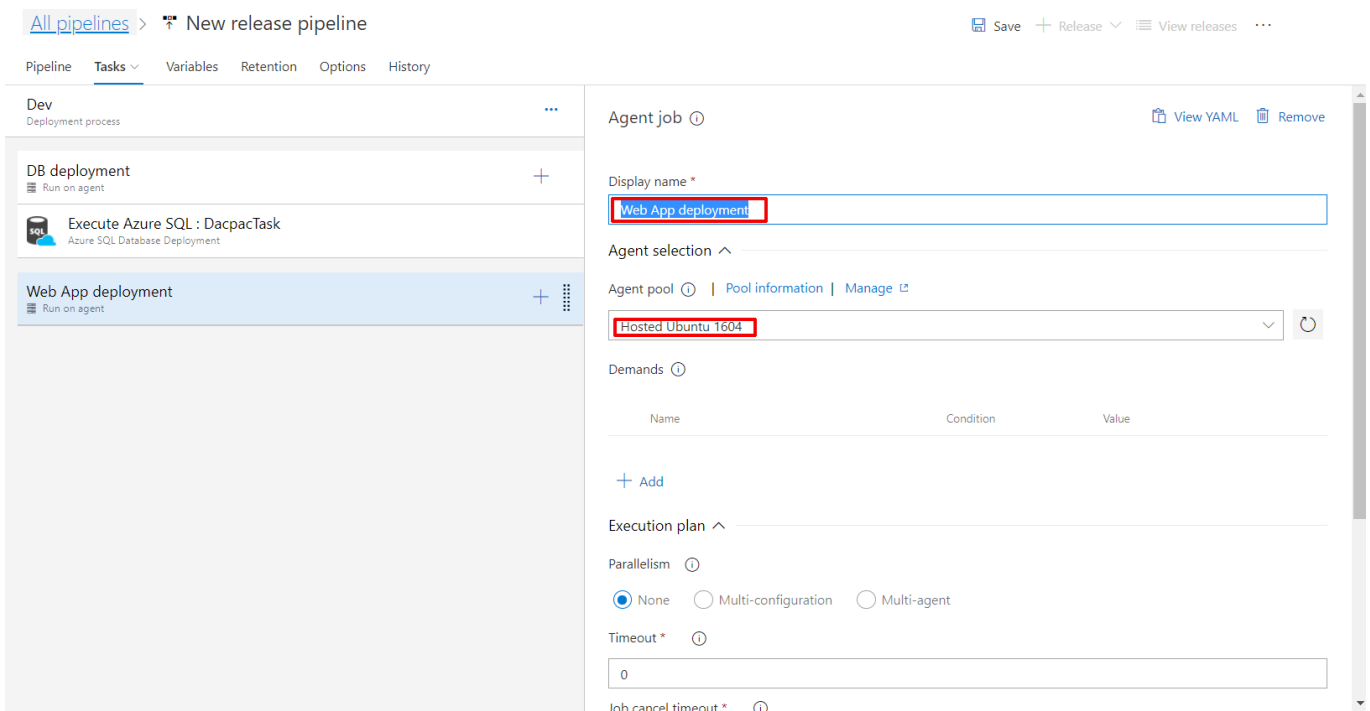
Variables list
SqlAzureDacpacDeployment1.SqlDeploymentOutputFile ①

Task 3: Config your Web App deployment

- Click on Ellipsis button (...) on **Dev** (Deployment process) and select **Add an Agent Job**



2. Click on your new **Agent job**
3. Set **Web App deployment** as your **Display name**
4. Select the **Hosted Ubuntu 1604** option from **Agent pool** dropdown



5. Click on plus button (+) and search for **Azure App Service Deploy**
6. Select the first one then click on **Add**

All pipelines > New release pipeline

Pipeline Tasks Variables Retention Options History

Dev
Deployment process

DB deployment
Run on agent

Execute Azure SQL : DacpacTask
Azure SQL Database Deployment

Web App deployment
Run on agent

Add tasks | Refresh

Azure App Service Deploy

Update Azure App Services on Windows, Web App on Linux with built-in images or Docker containers, ASP.NET, .NET Core, PHP, Python or Node.js based Web applications, Function Apps on Windows or Linux with Docker Containers, Mobile Apps, API applications, Web Jobs using Web Deploy / Kudu REST APIs

by Microsoft Corporation

Learn more

Marketplace

Azure App Service: Set App settings
Add settings to the App Settings of an Azure App Service.

Azure App Service: Add or remove IP Restrictions
Add or remove IP Restrictions, use custom address or add the IP of the build agent.

Octopus Deploy Integration
Build and Release tasks and other features for integrating with Octopus Deploy. Octopus is great for deploying ASP.NET or Core apps to on IIS or Azure, SQL databases, Windows services and much more.

Kudu ZipDeploy
Deploy a Zip package to the Azure App Service using the Kudu zipdeploy api

7. Click on your new **Azure App Service Deploy**:
8. Select **3.*** option from **Task version** dropdown
9. Set **Azure App Service Deploy** as your **Display name**
10. Select **ServiceConnection** on **Azure subscription**
11. Select **Linux web App** on **App type**
12. Select your **webapp** on **App Service Name**
13. Set **\$(ACR)** on **Registry or Namespace** input

All pipelines > New release pipeline

Pipeline Tasks Variables Retention Options History

Dev
Deployment process

DB deployment
Run on agent

Execute Azure SQL : DacpacTask
Azure SQL Database Deployment

Web App deployment
Run on agent

Azure App Service Deploy
Some settings need attention

Azure App Service Deploy

Task version 3.*

Display name *
Azure App Service Deploy

Azure subscription * | Manage
ServiceConnection

App type *
Linux Web App

App Service name *
[Redacted]

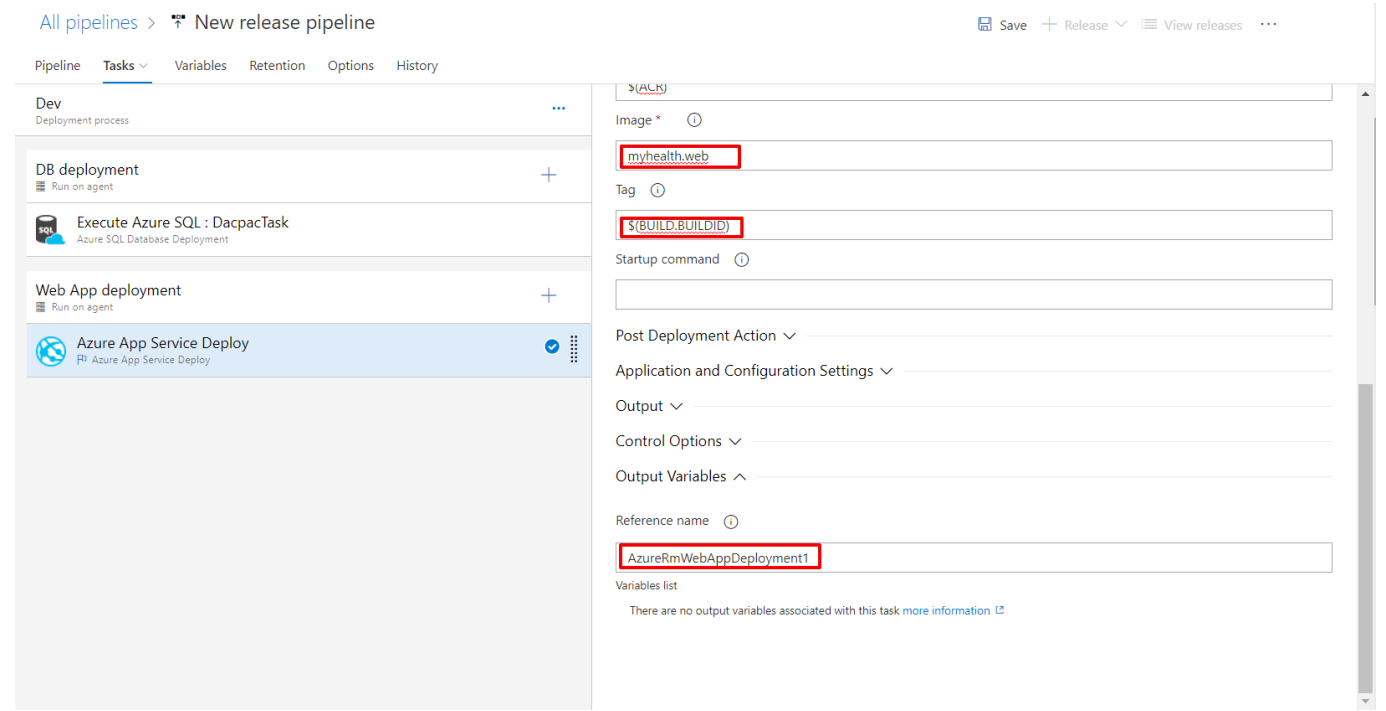
☐ Deploy to slot

Image Source
Container Registry

Registry or Namespace *
\$(ACR)

Image *

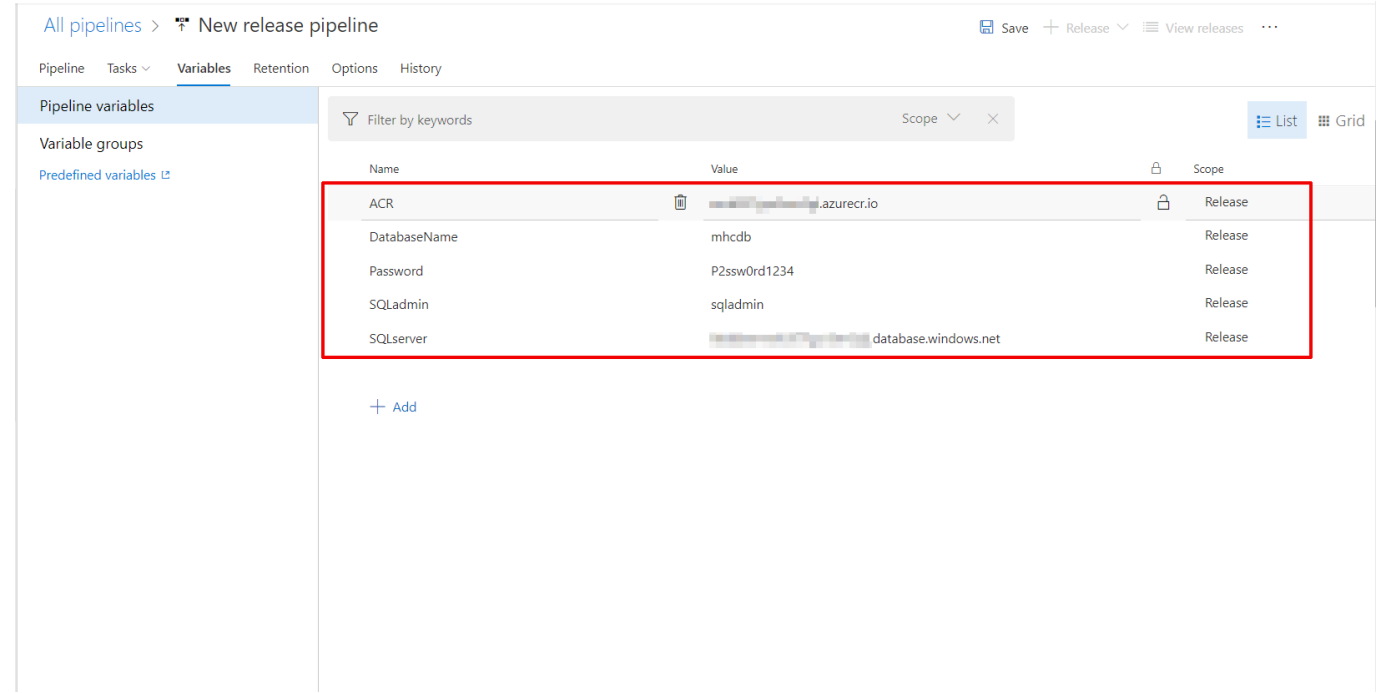
14. Put **myhealth.web** in the **Image** input
15. Put **\$(BUILD.BUILDID)** in **Tag** input
16. Put **AzureRmWebAppDeployment1** in **Reference name** on **Output Variables** section



Task 4: Config your variables

- 1. Click on **Variables** tab
- 2. Add these variables clicking on + **Add** button

Name: ACR **Value:** YOUR_ACR.azurecr.io **Scope:** Release **Name:** DatabaseName **Value:** mhcdb **Scope:** Release **Name:** Password **Value:** P2ssw0rd1234 **Scope:** Release **Name:** SQLadmin **Value:** sqladmin **Scope:** Release **Name:** SQLserver **Value:** YOUR_DBSERVER.database.windows.net **Scope:** Release



- 3. Click on **save** button

[All pipelines](#) > [New release pipeline](#)

[Save](#)
[Release](#)
[View releases](#)

[Pipeline](#)
[Tasks](#)
[Variables](#)
[Retention](#)
[Options](#)
[History](#)

[Pipeline variables](#)

[Variable groups](#)
[Predefined variables](#)

Filter by keywords Scope

Name	Value	Scope
ACR	██████████.azurecr.io	Release
DatabaseName	mhcdb	Release
Password	P2ssw0rd1234	Release
SQLadmin	sqladmin	Release
SQLserver	██████████.database.windows.net	Release

[+ Add](#)

Exercise 4: Initiate the CI Build and Deployment through code commit

1. Click on **Files** section under the **Repos** tab and navigate to the Docker/src/MyHealth.Web/Views/Home folder and open the Index.cshtml file for editing

[Azure DevOps](#)
/ DockerLab / Repos / Files / Docker

[DockerLab](#)

[Overview](#)
[Boards](#)
[Repos](#)
[Files](#)
[Commits](#)
[Pushes](#)
[Branches](#)
[Tags](#)
[Pull requests](#)

[Pipelines](#)
[Test Plans](#)
[Artifacts](#)

master Docker / src / MyHealth.Web / Views / Home / **Index.cshtml**

[Contents](#)
[History](#)
[Compare](#)
[Blame](#)
[Edit](#)
[Rename](#)
[Delete](#)
[Download](#)

```

1  @{
2      ViewData["Title"] = "HealthClinic";
3  }
4
5  <header>
6      <nav class="navbar navbar-default">
7          <div class="container-fluid mh-nav">
8              <div class="navbar-header">
9                  <button type="button" class="navbar-toggle hamburger collapsed" data-toggle=
10                     <span class="sr-only"></span>
11                     <span class="icon-bar"></span>
12                     <span class="icon-bar"></span>
13                     <span class="icon-bar"></span>
14                 </button>
15                 <a class="navbar-brand" href="/">
16                     
17                 </a>
18             </div>
19
20             <div class="collapse navbar-collapse" id="mh-menu">
21                 <div class="navbar-header visible-xs-block">
22                     
25
26                 <ul class="nav navbar-nav">
27                     <li class="active"><a href="#">HOME</a></li>
28                     <li><a href="#">JOIN US</a></li>
29                     <li><a href="#">TOUR INSIDE</a></li>
30                 </ul>
31
32                 <ul class="nav navbar-nav login-container hidden-xs">
33                     <li><a asp-controller="Account" asp-action="Login">Login</a></li>

```

2. Modify the text **JOIN US** to **CONTACT US** on the line number 28 and then click on the **Commit** button. This action would initiate an automatic build for the source code



3. After clicking **Commit** button, add a comment and click on **Commit**

A screenshot of a 'Commit' dialog box. It has a title bar with a close button (X). The dialog contains three input fields: 'Comment' with the text 'Updated Index.cshtml', 'Branch name' with the text 'master', and 'Work items to link' with a search bar containing the text 'Search work items by ID or title'. At the bottom right, there are two buttons: 'Commit' (blue) and 'Cancel' (gray).

4. Click on **Builds** tab, and subsequently select the commit name

Azure DevOps interface showing the 'Builds' tab for the 'MHCDocker.build' pipeline. The 'Builds' tab is highlighted in the left sidebar. The main view shows a table with one build entry: 'Updated Index.cshtml' (Commit) for build #20190209.2 on the master branch. The build is in a 'Queued' state.

#20190209.2: Updated Index.cshtml
Triggered today at 23:35 for DC Courses Docker master 586269e

Logs Summary Tests

Docker Job Started: 08/02/2019, 23:39:48
Pool: Hosted Ubuntu 1604 · Agent: Hosted Agent 3m 29s

Task	Status	Duration
Initialize job	succeeded	1s
Initialize Agent	succeeded	<1s
Checkout	succeeded	8s
Run services	succeeded	3m 4s

Build services 15s

```

Starting: Build services
=====
Task       : Docker Compose
Description: Build, push or run multi-container Docker applications. Task can be used with Docker or Azure Container registry.
Version    : 0.4.26
Author     : Microsoft Corporation
Help       : [More Information](https://go.microsoft.com/fwlink/?linkid=848006)
=====
[command]/usr/local/bin/docker-compose -f /home/vsts/work/1/s/docker-compose.yml -f /home/vsts/agents/2.146.0/.docker-compose.1549690983352.yml -p Docker build
Building myhealth.web
  
```

- The Build will generate and push the docker image of the web application to the Azure Container Registry. Once the build is completed, the build summary will be displayed.

Microsoft Azure

Search resources, services, and docs

Create a resource

Home

Dashboard

All services

FAVORITES

All resources

Resource groups

App Services

Function Apps

SQL databases

Azure Cosmos DB

Virtual machines

Load balancers

Storage accounts

Virtual networks

Azure Active Directory

Monitor

Advisor

Security Center

Cost Management + Billing

Azure services

See all (+100)

Virtual machines

Storage accounts

App Services

SQL databases

Azure Database for PostgreSQL

Azure Cosmos DB

Kubernetes services

Function Apps

Azure Databricks

Cognitive Services

Make the most out of Azure

Learn Azure with free online courses by Microsoft

Microsoft Learn

Monitor your apps and infrastructure

Azure Monitor

Secure your apps and infrastructure

Security Center

Optimize performance, reliability, security, and costs

Azure Advisor

Connect to Azure via an authenticated browser-based shell

Cloud Shell

Recent resources

See all your recent resources

See all your resources

NAME	TYPE	LAST VIEWED
mhwcbeapp	App Service	Just now
mhcdb (mhcdatabaseserver/mhcdb)	SQL database	40 min ago
acrmhc	Container registry	40 min ago
DockerRG	Resource group	2 h ago
MyVNETD	Virtual network	3 h ago
Pase para Azure: patrocinio	Subscription	3 h ago

Useful links

Get started or go deep with technical docs

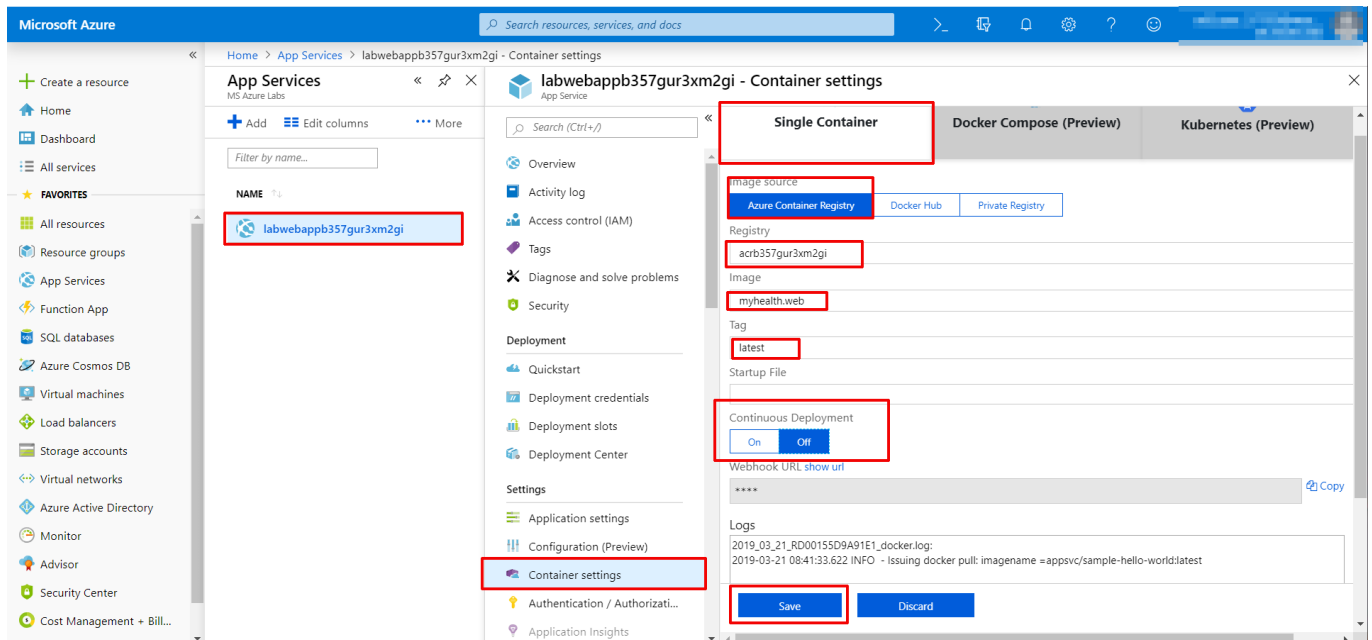
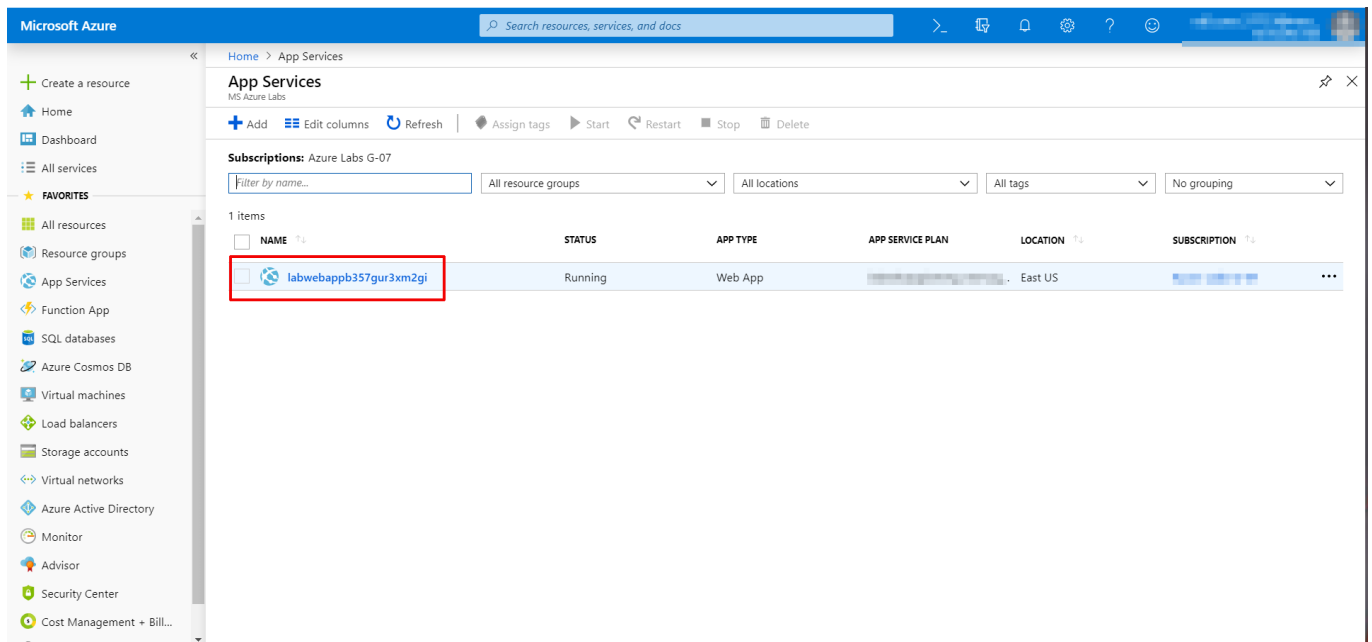
Our articles include everything from quickstarts, samples, and tutorials to help you get started, to SDKs and architecture guides for designing applications.

Discover Azure products

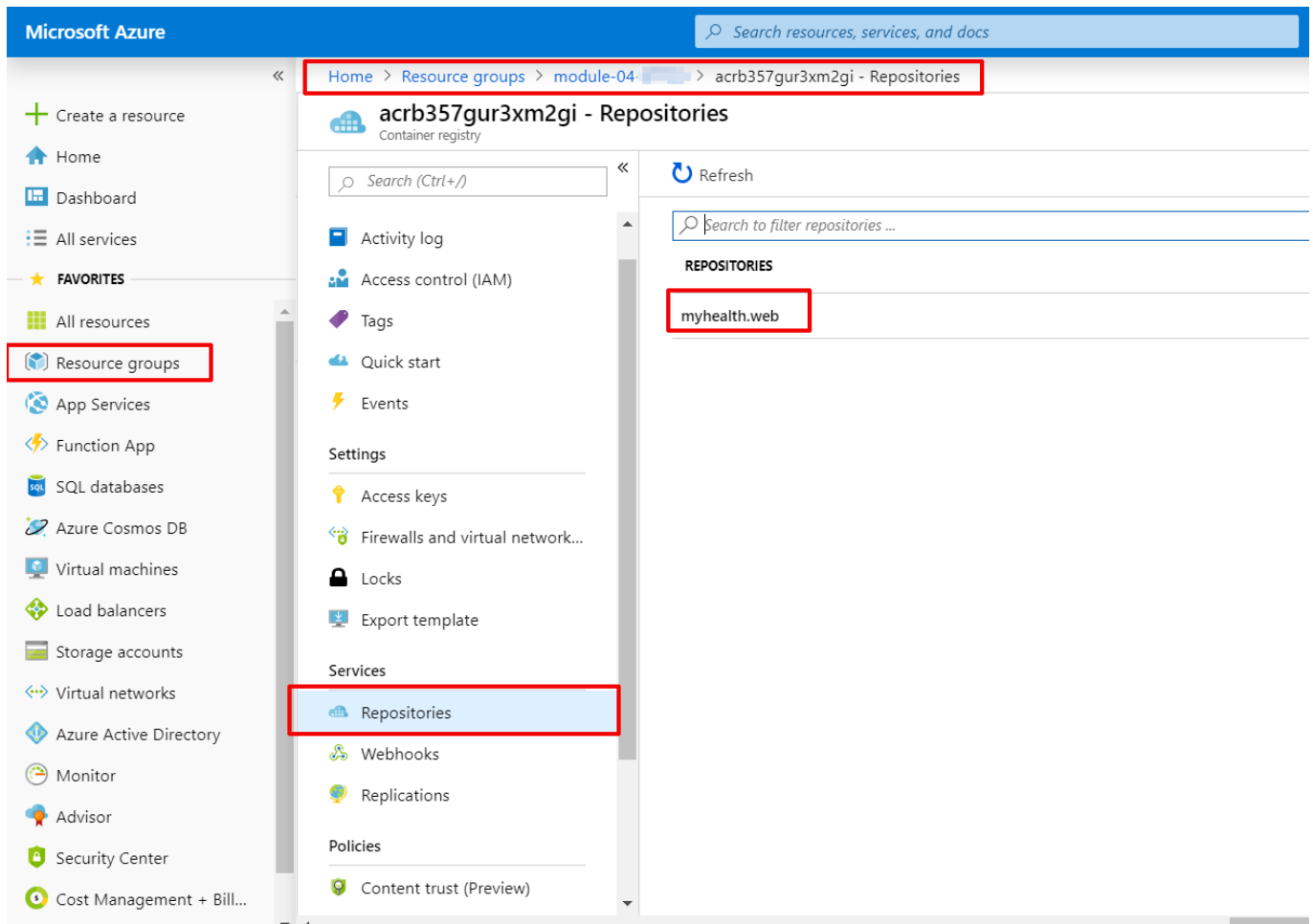
Explore Azure offers that help turn ideas into solutions, and get info on support, training, and pricing.

Keep current with Azure updates

Learn more and what's on the roadmap and subscribe to notifications to stay informed. Azure.Source wraps up all the news from last week in Azure.

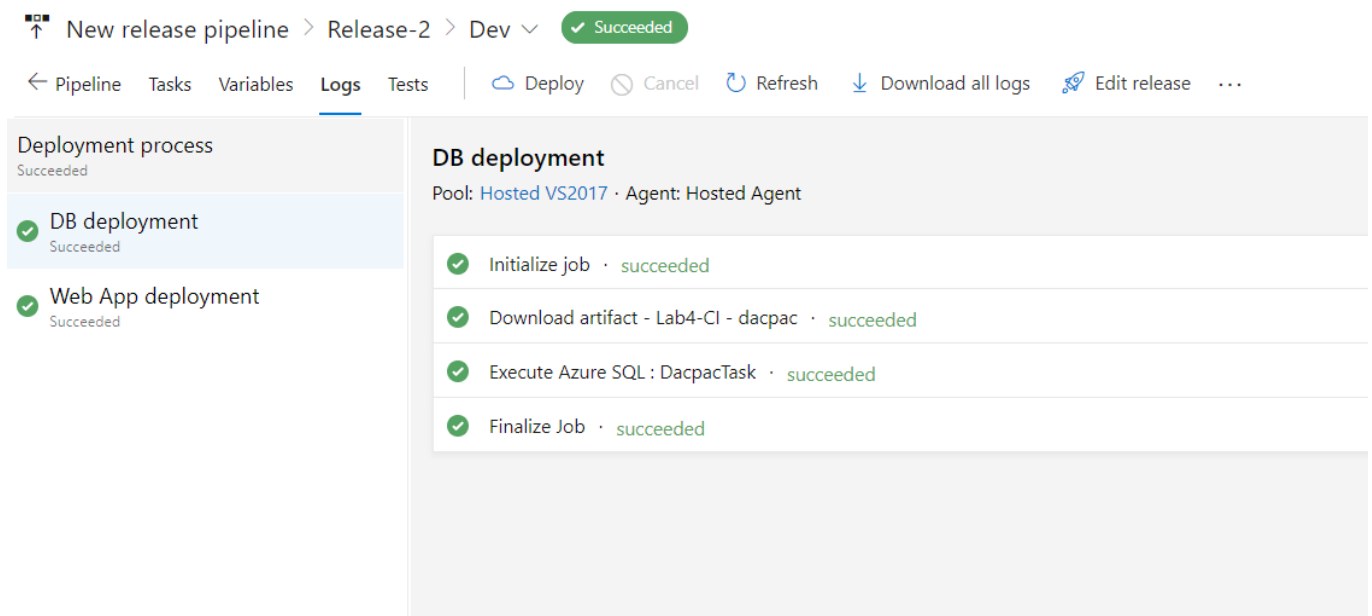


7. Navigate to the **Azure Container Portal** and then select the **Repositories** option to view the generated docker images



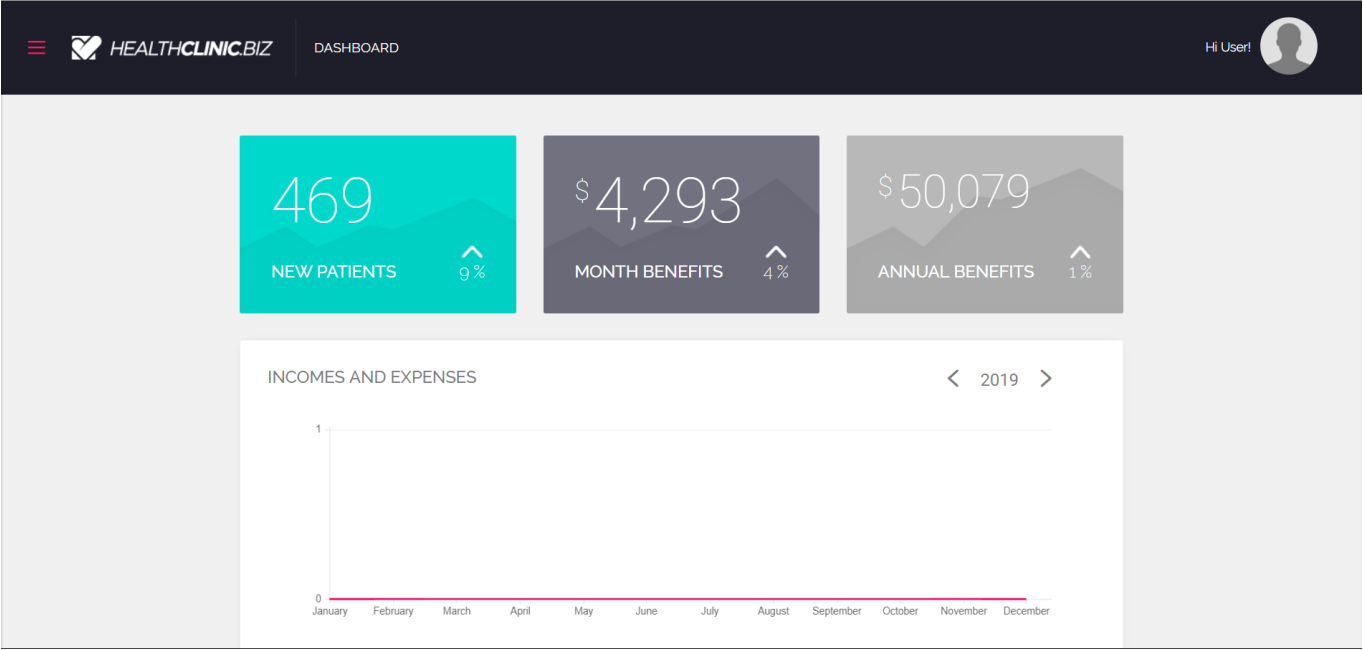
8. Navigate to the **Releases** section in **Azure DevOps** under **Pipelines** tab and double-click on the latest release displayed on the page. Click on Logs to view the details of the release in progress

note In case doesn't exist any release you can create a new one clicking on **create a release** and selecting the **Dev** from the pipeline



9. Navigate back to the [Azure Portal](#) and click on the **Overview** section of the **App Service**. Click on the link displayed under the **URL** field to browse the application and view the changes

10. Use the credentials **Username:** user and **Password:** P2ssw0rd@1 to login to the HealthClinic web application.



End of the lab